| **#** | **Area** | **Assumption** | **Remarks** |
| --- | --- | --- | --- |
| 1 | **Source System Access – Oracle (On-Prem)** | Access to the Oracle database or its staging output is available for delta comparison | Required for initial three-way reconciliation |
| 2 | **Source System Access – Read Replica (PostgreSQL)** | Access to the PostgreSQL read replica is available and consistent with production data | Access includes metadata for schema validation |
| 3 | **Source System Access – CDA (Cloud Data Access)** | Access to AWS S3 location containing CDA Parquet files is available | Includes read access via mount or access key-based integration |
| 4 | **Access Method to S3** | Databricks is configured to access S3 data via credential passthrough, secret scopes, or mount point | Required for automated validation of CDA files |
| 5 | **Staging Layer Availability** | A **staging layer for each source system** (Oracle, Read Replica, CDA) is made available in the Databricks environment | This will be the validation source for Bronze layer comparison |
| 6 | **Historical Access in Staging Layer** | Staging layers allow **access to 7 days of historical data** via versioning, partitioning, or timestamp-based querying | Enables reruns and traceable validations |
| 7 | **Delta Load Focus** | The validation scope is limited to **delta load only**, assuming historical data is already loaded and certified | Typically one-day loads used for validation |
| 8 | **Production Staging Readiness** | In **production**, all three staging layers (Oracle/Read Replica/CDA) will be present and accessible in a secure and stable manner | Enables post-deployment validation |
| 9 | **Workflow and DLT Access in Production** | Test automation team has the **required access in production Databricks** to create and manage **Workflows, Delta Live Tables (DLT), and Dashboards** | Needed for end-to-end validation reporting and orchestration |
| 10 | **DevOps Integration** | Azure DevOps is fully integrated for source control and CI/CD for test notebooks and configs | Supports version management and controlled promotion |
| 11 | **Table Scope Definition** | Only tables **approved and provided by the Data Engineering team** will be considered for test coverage | This ensures focus and agreement on target validation areas |
| 12 | **Test Data Provisioning** | Required test data for test execution will be **provided by the manual QA or UI automation team** | Automation team will not generate test data |
| 13 | **Schema Consistency** | The same schema structure is expected between Oracle, Read Replica, and CDA to allow automated schema validation | Critical to avoid false positives in structure comparison |
| 14 | **Firewall Limitations for JIRA** | Due to firewall restrictions, **Databricks cannot directly update self-hosted JIRA**; test metrics will be exported manually | Manual sync responsibility lies with the QA reporting team |
| 15 | **No Transformation Change Expected** | No new transformation logic is introduced during this migration; only ingestion source is changing | Reduces the need for complex transformation validations |
| 16 | **Silver/Gold Risk-Based Testing** | Only selected transformation-heavy tables will be validated based on **risk and impact**; reference tables will be fully covered | Ensures effort is focused on critical data flows |

**Section 5: Assumptions**

The assumptions below underpin the Automation Framework for Guidewire Data Migration validation. These assumptions outline key dependencies, system access, and data availability prerequisites necessary for successful test execution.

| **ID** | **Assumption** | **Description / Detail** |
| --- | --- | --- |
| **A1** | Source System Access | Automation team will have required read-only access to source systems: • **On-Prem Oracle database** (existing source) • **PostgreSQL Read Replica** (interim source) • **AWS S3 bucket (CDA files)** – through secured access methods provided by Data Engineering. |
| **A2** | Staging Layer Availability | A dedicated staging layer will be provided by the Data Engineering team for all source systems (Oracle, Read Replica, CDA Files). This staging layer is critical for automation validation and will support **7 days of historical data**, available via **version-controlled snapshots**. |
| **A3** | Databricks Access to S3 Data | Databricks environment will have secure and authorized access to AWS S3 buckets hosting CDA files through: • IAM roles and secure mounts (assumed configured by Cloud Ops/Data Engineering teams). • Direct mount points or dbfs mount operations allowing PySpark notebooks to read Parquet files directly. |
| **A4** | Delta Load Data Availability | Automation tests will focus exclusively on **incremental (delta) loads**, typically representing a **one-day data snapshot** from production or production-like environments. Historical data is excluded from validation scope. |
| **A5** | Approved Table Scope | Only tables explicitly approved and provided by the Data Engineering team will be considered within the automation testing scope. |
| **A6** | Risk-Based Approach | Validation within Silver and Gold layers will follow a risk-based testing approach: • **Transformation-heavy tables** – selectively tested based on risk assessment. • **Reference tables** – fully validated to ensure data integrity. |
| **A7** | Outbound Layer Accessibility | Databricks has read-only access to the outbound file system mount/storage locations for validation of file generation. No content-level validation is assumed or required. |
| **A8** | No Business Logic Changes | The migration effort is purely technical; no new transformation or business logic changes are anticipated. Existing known defects or functional logic validations are explicitly excluded from automation coverage. |
| **A9** | Test Data Provisioning | Test data required for validation scenarios will be prepared and provisioned by the **manual QA and UI automation teams**. Automation assumes this data will be present and correctly configured in the staging layer. |
| **A10** | Environment Readiness | Test and Production environments will be fully provisioned and ready for execution prior to starting automation tests. This includes: • Databricks clusters availability • Staging layer and data readiness • Appropriate permission levels |
| **A11** | Databricks Workflow & DLT Pipelines Access | Automation team will have permission to create, schedule, and manage Databricks Workflows and Delta Live Table (DLT) pipelines in both test and production Databricks environments. |
| **A12** | Dashboard Creation and Access | Automation team will have the necessary permissions to develop and maintain Databricks dashboards for result visualization in production and test environments. |
| **A13** | Version Control and Deployment | Azure DevOps is fully operational and integrated with Databricks, allowing the automation team to manage code versioning, Continuous Integration/Continuous Deployment (CI/CD), and release management effectively. |
| **A14** | Parallel Testing | Parallel testing of Read Replica and CDA ingestion methods is contingent on the simultaneous availability of both ingestion paths during the test execution window. Priority is given to Read Replica validation if parallel testing is not feasible. |
| **A15** | Firewall Constraints (JIRA Integration) | Due to firewall constraints between cloud-based Databricks and self-hosted JIRA, automated JIRA integration is not feasible. Manual reporting processes will be established and followed instead. |
| **A16** | Performance Benchmarking | Performance validation is limited to high-level ingestion timing comparisons between existing Azure Data Factory (ADF) pipelines and Databricks ingestion pipelines for a single-day load. Deep performance tuning and analysis are explicitly out of scope. |

## ****Section 7: Test Environment****

This section describes the environment landscape used for the development, execution, and validation of the automation framework designed to test data ingestion from Guidewire systems into Azure Data Lake. Each environment supports a distinct purpose and level of access to support a phased and controlled testing approach.

### **7.1 Environments in Scope**

| **Environment** | **Purpose** | **Scope of Access and Activities** |
| --- | --- | --- |
| **Development (DEV)** | To develop and validate the automation framework, including CI/CD pipelines. | - Used to test initial versions of automation scripts. - Validate integration between **Azure DevOps**, **Git**, and **Databricks Workflows**. - Validate connectivity to dummy/test data sources. - Full access to modify notebooks, workflows, and CI/CD configurations. - Simulate deployment lifecycle from Dev to Test. - Sandbox for onboarding new team members. |
| **Test (Non-Prod / QA)** | Primary environment for full automation test execution and validation. | - Full access to staging layers for **Oracle**, **PostgreSQL Read Replica**, and **CDA files** (via VPN-based S3 access). - Hosts **up to 7 days of delta load history** using version control. - Full validation coverage of Bronze, Silver, and Gold layers. - Dashboards, DLT pipelines, and notebooks can be created and managed directly by the automation team. - Enables **parallel testing** of Read Replica and CDA ingestion flows. |
| **Pre-Production / Production-like (DG / UAT)** | Used to validate automation behavior on near-production data and configurations. | - Contains **production-like data volume** and schema. - Access to read data from staging layers populated with production-refreshed data. - Validate high-level ingestion metrics, ADF vs Databricks load time comparisons. - Validates automation stability under realistic load volumes. - Used to ensure readiness before promoting code to Production. |
| **Production (PROD)** | Final environment for post-deployment validation and monitoring. | - Read-only access to production staging layers from **PostgreSQL Read Replica** and **CDA files via S3**. - Automation team has permission to execute **Databricks Workflows**, monitor **DLT pipelines**, and access **dashboards**. - Focused on validation of ingestion success, file generation in outbound layer, and performance benchmarks. - Limited scope: No changes to logic or critical pipelines permitted. |

### **7.2 Access Matrix by Environment**

| **Functionality** | **DEV** | **TEST** | **PRE-PROD** | **PROD** |
| --- | --- | --- | --- | --- |
| Create/Edit Databricks Notebooks | ✅ | ✅ | ❌ | ❌ |
| Create/Edit Databricks Workflows | ✅ | ✅ | ❌ | ✅ (with restrictions) |
| Trigger and Monitor DLT Pipelines | ✅ | ✅ | ✅ (read-only) | ✅ |
| Access Staging Layer (Oracle, RR, CDA) | ✅ | ✅ | ✅ | ✅ |
| Access 7-day Historical Data Snapshots | ❌ | ✅ | ✅ (if available) | ✅ (if available) |
| Execute Full Validation Suite | ✅ | ✅ | ⚠️ Partial | ⚠️ Limited (Post-deploy only) |
| Perform Parallel Ingestion Testing | ❌ | ✅ | ✅ (if both sources available) | ❌ |
| Compare ADF vs Databricks Load Time | ⚠️ Sample | ✅ | ✅ | ✅ |
| Dashboard Creation & Access | ✅ | ✅ | View Only | View Only |
| Validate File Generation (Outbound Layer) | ❌ | ✅ | ✅ | ✅ |
| CI/CD Pipeline Deployment via DevOps | ✅ | ✅ | ✅ | ✅ |

### **7.3 Notes and Special Conditions**

* **DEV** environment is used extensively to test the CI/CD pipelines, automation script releases, and deployment behavior before moving to TEST.
* **TEST** environment provides full access and flexibility for complete test coverage including schema, data, and performance validation.
* **PRE-PROD** serves as a sanity check before deployment, used mainly for production-like data volume validations and ingestion benchmarks.
* **PROD** access is tightly controlled. Focus is on **non-intrusive validation**, file generation checks, and ingestion monitoring.

**Section 9: Access Matrix by Environment**

This section defines the access levels required by the automation team across various environments involved in the Guidewire data ingestion validation process. It ensures that appropriate permissions are in place to support the execution, monitoring, and reporting of automation tests through the entire delivery lifecycle.

**9.1 Environment Access Overview**

| **Environment** | **Purpose** | **Automation Access Level** | **Key Activities Allowed** |
| --- | --- | --- | --- |
| **Development** (Dev) | Environment for validating CI/CD pipeline setup and deployment process. | **Read/Write** (full control of dev resources) | - Code integration testing - Notebook deployment validation - Databricks Workflow testing - CI/CD validation using Azure DevOps |
| **Test** (Non-Prod) | Primary execution environment for full functional automation validation. | **Full Access** | - Automated data validation (Bronze/Silver/Gold) - Workflow creation - Dashboard development - Delta Live Table (DLT) pipeline execution - Test reporting |
| **Pre-Production** (UAT/Stage) | Production-like environment for high-level benchmarking and smoke validation. | **Limited Access (Read + Execute)** | - Execution of approved notebooks/workflows - High-level comparison tests - Dashboard read access - No development or write permissions |
| **Production** | Live data validation post-deployment and monitoring of job health and ingestion integrity. | **Restricted Access (Read-Only with Execute)** | - Execute validated production workflows - Validate file generation in outbound feeds - Access dashboards and logs - No code or pipeline modification permitted |

**9.2 Access Rights by Component and Environment**

| **Component** | **Dev** | **Test** | **Pre-Prod** | **Production** | **Description** |
| --- | --- | --- | --- | --- | --- |
| **Databricks Notebooks** | ✅ Full access | ✅ Full access | 🔒 Execute only | 🔒 Execute only | Run automation logic and validations |
| **Databricks Workflows** | ✅ Create/Edit | ✅ Create/Edit | 🔒 Execute only | 🔒 Execute only | Orchestrate test execution |
| **Delta Live Tables (DLT)** | ✅ Create/Edit | ✅ Create/Edit | 🔒 Read only | 🔒 Read only | Store validation results |
| **Dashboards (Databricks)** | ✅ Full access | ✅ Create/Edit | 🔒 View only | 🔒 View only | Test reporting and visualization |
| **Azure DevOps (CI/CD)** | ✅ Full pipeline setup | ✅ Deployment only | 🔒 Deployment trigger | 🔒 Deployment trigger | Version control and release management |
| **Source System Access (Oracle)** | ✅ Dev masking | ✅ Test snapshot | 🔒 Read only | 🔒 Read only | JDBC access to staging |
| **Read Replica (PostgreSQL)** | ✅ Dev masked | ✅ Read/Write staging | 🔒 Read only | 🔒 Read only | Primary ingestion source |
| **CDA AWS S3 Files** | 🔒 No access | ✅ via VPN + IAM | ✅ via VPN + IAM | ✅ via VPN + IAM | Access Parquet files for validation |
| **Staging Layers (All Sources)** | ✅ Truncated | ✅ Full (7-day) | ✅ Limited (1-day) | ✅ Full (Live) | Pre-ingestion data snapshots |
| **Outbound Storage/Mounts** | 🔒 No access | ✅ Read access | ✅ Read access | ✅ Read access | Validate file presence post-ingestion |

**9.3 Notes**

* **Databricks Cluster Access**: All environments must provide Databricks cluster access with appropriate compute permissions.
* **Role-Based Access Control (RBAC)** is enforced in production environments to restrict changes, ensuring audit compliance.
* **Deployment Pipeline**: Code is promoted progressively (Dev → Test → Pre-Prod → Prod) via Azure DevOps CI/CD process.
* **VPN-Based S3 Access**: Available only in **Test, Pre-Prod**, and **Production** environments, managed by the Cloud Ops and Security teams.
* **Firewall Rules and Network Access**: Configured by Infra/Security teams; any cross-network access (e.g., from Databricks to JIRA or internal sources) must be handled with approval and controlled routing.

Would you like to proceed to the final compilation of the full test plan or add any more custom sections (like visual diagrams, approval sign-off sheet, or test data workflow)?

### ✅ **Test Data Readiness Checklist (Pre-Execution)**

| **Item** | **Responsibility** | **Status** |
| --- | --- | --- |
| ✅ Test scenarios finalized and approved | Manual QA / UI Team | ☐ |
| ✅ Required test data created in Guidewire UI | Manual QA / UI Team | ☐ |
| ✅ Oracle data staged and accessible | Data Engineering | ☐ |
| ✅ Read Replica data staged and accessible | Data Engineering | ☐ |
| ✅ CDA files available in AWS S3 and accessible via VPN | Data Engineering / Cloud Ops | ☐ |
| ✅ 7-day historical snapshot partitioned in staging | Data Engineering | ☐ |
| ✅ Databricks workflow configured and scheduled | Automation Team | ☐ |
| ✅ Dashboards connected to result views | Automation Team | ☐ |
| ✅ Manual JIRA update process prepared | QA Coordinator | ☐ |

## ****Section 11: Test Activities and Deliverables****

This section outlines all key activities, processes, and deliverables associated with the test automation effort for the Guidewire data migration project. It ensures alignment across QA, automation, and data engineering teams, and supports traceable execution and outcome reporting.

### **11.1 Test Design**

* **Objectives**:
  + Design validation logic aligned to data migration architecture (Bronze, Silver, Gold).
  + Identify test conditions for schema, data, SCD, performance, and anomaly validations.
  + Build reusable PySpark validation modules within Databricks notebooks.
* **Deliverables**:
  + Test case matrix mapped to data sources and layers
  + SQL/PySpark logic repository
  + Automated notebook structure (parameterized)

### **11.2 Test Execution**

* **Automated Execution Flow**:
  + Triggered via **Databricks Workflows** post ingestion.
  + Executes validation logic against staging and Delta layers.
  + Stores output in Delta Live Tables and dashboard views.
* **Execution Logs & Reports**:
  + Stored in Delta format for traceability
  + Captured for each cycle and date range

### **11.3 Entry Criteria for Test Execution**

| **Entry Checkpoint** | **Required Status** |
| --- | --- |
| Test scenarios reviewed and signed off | ✅ |
| Required test data created and present in staging layers (Oracle, Read Replica, CDA) | ✅ |
| Databricks Workflows and DLT pipelines created and tested | ✅ |
| Dashboards connected to result views | ✅ |
| Environment access for automation and validation is confirmed | ✅ |
| CDA S3 VPN access tested and available | ✅ |

### **11.4 Exit Criteria for Test Execution**

| **Exit Checkpoint** | **Expected Outcome** |
| --- | --- |
| All in-scope test cases executed successfully | ✅ |
| No **Critical/High** severity defects open | ✅ |
| Test evidence stored and accessible | ✅ |
| Summary reports and dashboards reviewed | ✅ |
| Defect retesting completed and closed | ✅ |
| Final results communicated and accepted by stakeholders | ✅ |

### **11.5 Test Cycles**

* **Cycle 1 – Read Replica Validation**
  + Execute all test cases for ingestion from PostgreSQL Read Replica.
  + Includes comparison with Oracle and Bronze layers.
* **Cycle 2 – CDA Validation**
  + Execute CDA file ingestion validation (Parquet from AWS S3).
  + Includes three-way comparison with Oracle, Read Replica, and CDA.
* **Cycle 3 – Silver and Gold Risk-Based Testing**
  + Transformation checks for selected tables.
  + Reference table full coverage validation.
* **Cycle 4 – Parallel Ingestion Validation (If Available)**
  + Validate both ingestion methods (Read Replica & CDA) for parity.

### **11.6 Test Schedule**

| **Phase** | **Tentative Duration** | **Activities** |
| --- | --- | --- |
| Test Planning | Week 1 | Finalize scenarios, access setup |
| Test Design | Week 2 | Notebook development, CI/CD setup |
| Test Execution Cycle 1 (Read Replica) | Week 3 | Validate interim ingestion |
| Test Execution Cycle 2 (CDA) | Week 4 | Validate final ingestion |
| Test Execution Cycle 3 (Silver/Gold) | Week 5 | Risk-based transformation validation |
| Final Report & Handoff | Week 6 | Sign-off and reporting |

Note: Dates may shift based on CDA readiness and environment provisioning.

### **11.7 Periodic Progress and Metrics Reporting**

* **Frequency**: Twice per week (or daily during active execution)
* **Medium**: Databricks Dashboards, Excel/CSV export, JIRA updates
* **Metrics Tracked**:

# of test cases executed/passed/failed

# of defects logged/resolved

* + Validation status by layer/source
  + SLA adherence (execution, defect turnaround)

### **11.8 Defect Tracking and Management**

| **Activity** | **Tool/Process** | **Owner** |
| --- | --- | --- |
| **Defect Logging** | Manual entry in JIRA (due to firewall limits) | QA Lead / Validator |
| **Defect Triage** | Daily review and classification (High, Medium, Low) | QA + Data Engineering |
| **Defect Assignment** | Based on source (ingestion, transformation, or reference) | QA/Dev Lead |
| **Defect Retesting** | Post fix re-validation using automation | Automation Team |

### **11.9 Test Evidence and Documentation**

| **Evidence Type** | **Location** | **Generated By** |
| --- | --- | --- |
| SQL/PySpark Queries Used | Git (Azure DevOps) | Automation Framework |
| Validation Results | Delta Tables (Databricks) | Databricks Workflows |
| Dashboards | Databricks Views | Automation Framework |
| Execution Logs | Delta/DBFS | Databricks |
| JIRA Reports | Manual update | QA Team |

## ****Section 11: Test Activities and Deliverables****

This section outlines the end-to-end testing lifecycle activities, key deliverables, and quality control checkpoints that will govern the automation testing of Guidewire data ingestion into Azure Data Lake. The activities are organized into logical sub-sections to manage scope, execution, quality, and traceability.

### **11.1 Test Design**

| **Activity** | **Description** |
| --- | --- |
| **Test Scenario Mapping** | Identify automation-suitable scenarios from agreed table list and ingestion flows (Oracle, Read Replica, CDA). |
| **Test Case Development** | Define automated test cases for schema validation, count checks, data accuracy, referential integrity, performance, etc. |
| **SQL/Logic Generation** | Develop reusable SQL/PySpark code templates for validations. |
| **Risk-Based Coverage Planning** | Prioritize tables for Silver/Gold validation using risk-based classification. |
| **Baseline Reporting Templates** | Prepare standard reporting format for dashboards and logs. |

### **11.2 Test Execution**

| **Activity** | **Description** |
| --- | --- |
| **Automation Triggering** | Execute Databricks workflows for each ingestion source (Read Replica / CDA). |
| **Validation Run** | Schema, record count, field comparison, duplicate, SCD checks, etc., performed layer-wise. |
| **Result Storage** | Write results to Delta Live Tables and generate result views for reporting. |
| **Parallel Validation** | Run parallel validations for Read Replica and CDA if both are ready. |
| **Manual Update to JIRA** | Export test run metrics and manually log them into JIRA due to network restrictions. |

### **11.3 Entry Criteria for Test Execution**

* Automation scripts/code reviewed and merged into main DevOps branch.
* Staging layers available and contain 1-day delta load.
* Access to Oracle, Read Replica, and CDA S3 confirmed via secure methods.
* Required cluster resources available in Databricks.
* Test scenarios and data provisioned and confirmed by QA/UI team.

### **11.4 Exit Criteria for Test Execution**

* All planned test cases executed for in-scope tables and ingestion methods.
* No high or critical priority defects open related to ingestion validation.
* Results published in dashboard and exported for reporting.
* Stakeholder review of test summary and sign-off received.

### **11.5 Test Cycles**

| **Cycle** | **Purpose** | **Scope** |
| --- | --- | --- |
| **Cycle 1 – Dev Validation** | Dry run to validate workflows and scripts | 5–10 critical tables |
| **Cycle 2 – Full Non-Prod** | Execute all in-scope tables in test environment | All ingestion types |
| **Cycle 3 – Parallel Testing** | Read Replica vs CDA reconciliation (if both available) | Shared tables |
| **Cycle 4 – Production Validation** | Post-deployment sanity validation | Key Bronze/Silver/Outbound tables only |

### **11.6 Test Schedule**

| **Phase** | **Timeline** | **Deliverables** |
| --- | --- | --- |
| Test Design | [Start Date] – [End Date] | Test cases, SQL scripts, logic repository |
| Framework Setup | [Start Date] – [End Date] | DevOps pipeline, DLT setup, dashboards |
| Cycle Execution | [Start Date] – [End Date] | Execution reports, logs |
| Final Review & Sign-Off | [End Date] | Final defect summary, test summary |

(Insert actual dates as per project plan)

### **11.7 Periodic Progress & Metric Reporting**

| **Frequency** | **Report Type** | **Distribution** |
| --- | --- | --- |
| Daily | Execution Status (Dashboard + Email Summary) | Automation Lead, QA Manager |
| Weekly | Metrics Snapshot + Defect Overview | PMO, QA Leadership, Engineering |
| End of Cycle | Final Test Summary Report | Stakeholders, Business |

### **11.8 Defect Management**

#### **Defect Tracking**

* Logged manually into JIRA with severity, environment, and source information.
* Linked to the specific test case or validation type.

#### **Defect Triaging**

* Conducted jointly with QA, Data Engineering, and Automation teams.
* Categorize as schema/data issue, pipeline error, or transformation mismatch.

#### **Defect Retesting**

* Once fixed, rerun automation scripts against updated data.
* Re-validate expected vs actual results and mark the defect accordingly.

### **11.9 Test Evidence & Documentation**

| **Evidence Type** | **Content** | **Storage** |
| --- | --- | --- |
| Validation Results | Schema diffs, row mismatches, duplicate records, etc. | Delta Tables + Views |
| SQL/Code Snapshots | Queries used in test validation | Git (Azure DevOps) |
| Execution Logs | Run history, error logs, runtime metrics | Databricks logs / dashboard |
| Test Reports | Cycle-wise and summary dashboards | Databricks Dashboard / Excel export |
| JIRA Metrics | Manually exported test case results | Shared via Confluence / Tracker |

Let me know if you'd like this bundled into your full Test Plan, or if you want supporting templates (Excel-based trackers or dashboard wireframes).

## ****Section 12: Access Matrix by Environment – Strategy & Prerequisites****

This section outlines the environment-wise **strategy, purpose, access levels, and prerequisites** for implementing and executing the automation test framework in the Guidewire Data Migration project. Each environment plays a specific role in ensuring quality validation, safe deployment, and structured CI/CD pipeline management.

### **12.1 Environment Usage Strategy Overview**

| **Environment** | **Primary Purpose** |
| --- | --- |
| **Development (DEV)** | Validate the CI/CD pipeline, test deployments, and integration of automation framework components. |
| **Test (QA/UAT)** | Execute full-scale automation, functional validation, parallel source testing, and result reporting. |
| **Pre-Production (Pre-Prod)** | Conduct high-level sanity checks, performance comparisons, and smoke validations using production-like data. |
| **Production (PROD)** | Perform post-deployment ingestion validations, monitor job execution, and verify file generation. |

### **12.2 Access Matrix and Environment Prerequisites**

| **Component** | **DEV** | **TEST** | **PRE-PROD** | **PRODUCTION** |
| --- | --- | --- | --- | --- |
| **Databricks Workspace Access** | Full (admin-level for automation team) | Full (run, create workflows, DLT) | Execute-only | Execute-only |
| **Cluster Provisioning** | Dev-grade clusters with limited capacity | QA-grade clusters (scalable for full validation) | Similar to production (for comparison) | Prod-grade clusters with restricted controls |
| **Workflow Creation & Execution** | ✅ Create, edit, test deployment logic | ✅ Create, execute full validations | 🔒 Execution only | 🔒 Execution only |
| **Delta Live Table (DLT) Access** | ✅ Create and modify pipelines | ✅ Full use for validation logging | 🔒 Read-only | 🔒 Read-only |
| **Dashboard Creation & Management** | ✅ Create and modify dashboards | ✅ View and customize results | 🔒 View only | 🔒 View only |
| **CI/CD Deployment via Azure DevOps** | ✅ Test and validate CI/CD deployments | ✅ Receive and execute released packages | ✅ Pull from DevOps pipeline | ✅ Final deployment from master branch |
| **Source Data Access (Oracle)** | 🔒 Dev-masked or static snapshot | ✅ 1-day delta data in staging | 🔒 Read-only snapshot | 🔒 Read-only for 1-day verification |
| **Source Data Access (Read Replica)** | ✅ Basic for script testing | ✅ Full JDBC for ingestion validation | ✅ Read-only for comparison | ✅ Read-only, real-time delta access |
| **Source Data Access (CDA from S3)** | 🔒 Not configured | ✅ Full access via VPN & IAM | ✅ VPN-based access for test | ✅ VPN-based access to live files |
| **Staging Layer Access** | 🔒 Truncated test data | ✅ Full (7-day snapshot and version control) | ✅ 1-day production-like data | ✅ Live ingestion layer (1-day only) |
| **Outbound Feed Access** | 🔒 No access | ✅ Verify file creation only | ✅ File availability checks | ✅ Final outbound feed confirmation |
| **JIRA/Defect Management** | ✅ Dev-linked (manual only) | ✅ Manual test evidence updates | ✅ Summary reporting only | ✅ Summary reporting only |
| **Test Data Management** | 🔒 Not in scope | ✅ Coordinated with manual/UI team | 🔒 Not in scope | 🔒 Not in scope |

### **12.3 Environment-Specific Prerequisites**

#### 🧪 **Development (DEV)**

* CI/CD integration validated for Databricks + Azure DevOps
* Git branching strategy in place (feature, dev, release)
* Test repositories and parameters mocked or stubbed
* Admin access to Databricks workspace and Git repository

#### 🔍 **Test (QA/UAT)**

* Access to all ingestion sources (Oracle, Read Replica, CDA)
* 7-day version-controlled staging layers provisioned
* Full cluster compute availability for large dataset validation
* Databricks Workflows, DLTs, Dashboards pre-configured
* Manual team has completed test scenario and data setup

#### 📊 **Pre-Production (Pre-Prod)**

* Realistic production-like staging data for 1-day delta
* Databricks VPN connectivity to AWS S3 buckets validated
* Only essential test cases executed (high-impact coverage)
* No write permission to pipelines or clusters

#### 🚀 **Production (PROD)**

* Final deployment approved through Azure DevOps release pipeline
* Only automation workflows previously validated are executed
* Read-only access to staging, Bronze, Silver layers
* DLT pipelines must be deployed and locked before execution
* File presence validated in outbound folders (no content parsing)
* Monitoring dashboards in read-only mode for execution review

## ****Section 6: Automation Working Approach****

This section provides a comprehensive overview of the detailed working approach of the automation testing framework designed to validate the Guidewire data migration into Azure Data Lake.

### **6.1 Automation Framework Overview**

The automation framework is built upon **Azure Databricks**, leveraging **PySpark Notebooks** orchestrated through **Databricks Workflows**. The entire automation workflow is version-controlled, maintained, and deployed via **Azure DevOps** to ensure traceability, reliability, and repeatability.

The validation pipeline involves orchestrated workflows that perform schema validation, data accuracy checks, completeness checks, and high-level performance benchmarking, with results stored in Delta Lake and surfaced via dashboards for reporting and analysis.

### **6.2 Automation Working Steps**

#### **Step 1: Environment Readiness and Access**

* Ensure Databricks environments (**Test and Production**) are provisioned.
* Verify required cluster configurations and permissions are established.
* Confirm that all necessary source system access permissions are available, including:
  + **Oracle Database**
  + **PostgreSQL Read Replica**
  + **AWS S3 CDA Files** (via VPN-based access)
* Staging layer setup for each source (Oracle, Read Replica, CDA) with agreed retention policies (**7 days historical snapshots**).

#### **Step 2: Method of Accessing CDA Files from AWS S3**

* The CDA files hosted in AWS S3 buckets will be accessed directly via **Databricks Clusters** using **cluster-restricted VPN-based access** provided and managed by the Data Engineering and Cloud Operations teams.
* Data Engineering team will configure IAM roles, VPN tunnels, and mount points on the Databricks clusters, enabling secure, seamless, and direct access to CDA Parquet files in S3.

#### **Step 3: Source Data Extraction and Preparation**

* Automation scripts will connect securely to staging layers:
  + Oracle Database via JDBC connections.
  + PostgreSQL Read Replica databases via JDBC connections.
  + CDA S3 files via direct mounts on Databricks clusters (VPN-based).
* Extraction scripts will perform initial data ingestion into temporary Delta tables within Databricks for subsequent validation.

#### **Step 4: Validation Execution Workflow**

* **Orchestration**:  
  Automation scripts are executed within scheduled **Databricks Workflows**.
* **Execution Phases**:
  + Schema validation: Comparing source and Bronze schemas.
  + Data accuracy and consistency checks (row count, duplicate detection, SCD checks, and field-level comparisons).
  + Referential integrity checks based on primary/foreign keys.
* **Parallel Testing**:
  + When both Read Replica and CDA sources are simultaneously available, parallel validation ensures consistency between the two ingestion methods.
  + Otherwise, priority is explicitly placed on validating the Read Replica source first.

#### **Step 5: Results Processing and Dashboarding**

* Validation results are written to **Delta Live Tables (DLT)**.
* Results are transformed into summary views within Databricks.
* Dashboards built on these views provide visual analytics, validation outcomes, and status tracking in near real-time.

#### **Step 6: High-Level Performance Benchmarking**

* Automation scripts perform **basic ingestion timing comparisons** between existing Azure Data Factory (ADF) pipelines and new Databricks ingestion pipelines, based on a single day's worth of data.
* Metrics captured include:
  + Total ingestion runtime
  + Record count comparisons
  + Performance differences across pipelines

#### **Step 7: Reporting and JIRA Integration**

* Due to firewall constraints, automatic updates to JIRA are unavailable.
* Validation results, logs, and reports are exported from Databricks in CSV/Excel formats.
* QA/Test coordinators manually update these results into JIRA.

### **6.3 Automation Framework Tooling and Integration**

| **Component** | **Technology/Tool** | **Description** |
| --- | --- | --- |
| **Execution Environment** | Azure Databricks | Executes PySpark notebooks for data validation. |
| **Data Sources** | Oracle, PostgreSQL Read Replica, CDA via S3 | Source system connections for data extraction. |
| **VPN Access (CDA)** | Cluster-restricted VPN | Secure direct Databricks-to-S3 access provided by Data Engineering. |
| **Code Versioning** | Azure DevOps Git | Manages automation scripts and workflow versions. |
| **Workflow Orchestration** | Databricks Workflows | Manages scheduled execution and dependencies. |
| **Data Storage** | Delta Lake (Delta Tables & Views) | Stores and manages validation results and logs. |
| **Reporting and Analytics** | Databricks Dashboards | Visualization of test results and validation metrics. |
| **Performance Comparison** | Databricks & Azure Data Factory (ADF) | High-level ingestion pipeline timing benchmarks. |
| **Test Management** | JIRA (Manual updates) | Tracking and reporting manual validation results. |

### **6.4 Automation Scheduling and Monitoring**

* Scheduled execution of validation workflows occurs via Databricks workflow scheduling.
* Execution logs and metrics are continuously monitored via dashboards and Delta views.
* Issues identified during automation execution are logged in dashboards and manually entered into JIRA.

Let me know if you'd like to continue with additional sections or need further customization to this working approach.